PATENT SPECIFICATION

(11) 1 264 066

1 264 066

DRAWINGS ATTACHED

(21) Application No. 30732/70 (22) Filed 24 June 1970

(45) Complete Specification published 16 Feb. 1972

(51) International Classification G 06 f 9/16

(52) Index at acceptance
G4A 12C 12D 12N 13M 15A2 16D 16E2 16EY 16J 1F 2C 2F5 5A 6G 9C

(72) Inventors PHILIP CHARLES AUSTIN, ROBERT WILLIAM TAYLOR and MICHAEL HASLAM

(54) DATA PROCESSING APPARATUS

(71) We, INTERNATIONAL BUSINESS MACHINES CORPORATION, a Corporation organized and existing under the laws of the State of New York in the United States of America, of Armonk, New York 10504, United States of America, do hereby declare the invention for which we pray that a patent may be granted to us, and the method by which it is to be performed, to be particularly described in and by the following statement:—

The invention relates to data processing apparatus.

Considerable time and expense is incurred in debugging programs for data processing apparatus. To assist the programmer in debugging programs the contents of various registers may be dumped, after a program check has been detected. However, while the contents of a dump are helpful, it is not possible to trace back to the point in the program where the actual programming error occurred.

The present invention provides data processing apparatus for executing a program of instructions, in which predetermined situations occurring during the execution of an instruction are designated program events, the apparatus including means for indicating that a particular program event or events is to be detected, means for detecting that an indicated event has occurred, means responsive to the detection of an indicated event for recording that the indicated event has occurred, means for completing the execution of the instruction during which the indicated program event was detected and microprogram control means for fetching instructions from storage and for executing instructions in an arithmetical and logical unit, in which each program event has an associated microinstruction, and in which the detecting means is responsive to the indication that a particular event is to be detected and to the occurrence of the microinstruction associated with that event.

Thus the invention enables the programmer to trace the point at which an error occurred by recording certain specified program events. Such program events (PE) include:

1. the fetching of an instruction from designated main storage locations;
2. the successful execution of a branch instruction;
3. the alteration or use of designated general registers;

and

4. the alteration or use of designated main storage locations.

This is called herein "program event recording" (PER). While only four events are specified above many other events could be detected and recorded. The programmer can specify one or more events to be recorded during the execution of the same instruction. The information concerning a program event can be obtained during a program interruption with the cause of the interruption being specified by the interruption code. The program interruption is taken after the instruction in which the indicated program event is detected has been executed. In one embodiment of the invention a " program event detected " latch is set when an indicated event is detected and information concerning the event is set in a buffer. At the next instruction fetch time, the program event detected latch is tested and if set the program interrupt is taken to record the event.

The data processing apparatus may include means for inhibiting the operation of the program event indicating means whereby the apparatus may operate without program event recording being available to the programmer. The inhibiting means may include a program event mode indication bit located in a program status word (PSW) used to control the apparatus.

In a microprogram controlled data processing apparatus each program event may be detected when a microinstruction associated with that event is used to control the data processing apparatus. When the program event is detected a microprogram subroutine can be used to set the program event detected latch and to store information relating to the pro-

gram event in a buffer. Control of the data processing apparatus then returns from the subroutine to the main microprogram to complete the execution of the instruction in which the program event had been detected. When the next Instruction fetch microinstruction is processed the program event detected latch is tested and if set the program interrupt is taken to record the program event.

Each program event to be detected may be indicated by setting a program event code latch. The detection of a program event can be achieved by ANDing the set output of a program event code latch with the appropriate bits of the operations code of the microinstruction. In addition to providing means for inhibiting program event recording, the output of the event code latch and the appropriate bits of the operations code of the microinstruction may be ANDed together with the program event mode bit located in the program status word.

Program events which involve a main store address comparison or general purpose register can be very numerous. In addition information stored in various registers must be protected after a program event has been detected while the microprogram sets the program event detected latch and stores information relating to the event. In one embodiment of the invention data is logged-out into dedicated areas of the apparatus. The program event can then be evaluated using this dedicated area. The evaluation determines if the storage location is in a designated area of store or if the register is a designated register. The data logged out includes the contents of a microinstruction address register, main store address register, instruction counter and various general purpose registers. The dedicated area may be a dedicated area normally reserved for handling errors or system stop conditions. In an embodiment of the invention a local store is provided in which one portion is set aside for normal processing and another portion is set aside to act as the dedicated area for log-out.

The invention will now be described by way of example with reference to the drawing which shows a microprogram controlled data processing apparatus in which the present invention is embodied.

The data processing apparatus includes a storage unit 1 comprising a main store (MS) for storing operands and instructions, and a control store for storing control words. The control words or microinstructions control the operation of the apparatus. The microinstruction currently controlling the operation of the apparatus is located in control data register 2 (CDR). The contents of the CDR are decoded by decoder 3 and control lines 4 transmit control signals to various parts of the system.

Storage address register 5 (SAR) is used for accessing words of 16 data bits from the main and control stores. Buffer address register 6 (BAR) and incrementer 7 store and increment respectively the address in SAR. An address may be entered into SAR from BAR, Work Store 8 or CDR. Alternatively, an address may be forced or a digit may be entered directly from Work Store 8. Two local stores are provided, Work Store 8 which consists of eight separate zones; and an auxiliary store 9 which consists of sixteen general purpose registers (GPR), 8 floating point registers (FPR) and 8 common work registers (CWR). Operands processed by arithmetic and logical unit 10 (ALU) are stored in the work store 8, auxiliary store 9 or can be obtained directly from CDR by means of the K bus 10*a*.

Each zone of work store 8 is allocated a particular function, Zone O is allocated to CPU operation, that is, all instruction fetches and all instruction executions except those concerned with input-output operations, Zone 1 to system reset, logout, retry and program event recording (PER), Zone 2 to low speed input/output (I/O) device, Zone 3 to a printer, Zone 4 to communication devices, Zone 5 to disc files and Zones 6 and 7 to selector channels. Only one zone is actively involved in a microprogram task at any one time. A change of task is carried out during a microprogram interrupt cycle. The active zone is selected by select latches 11. The zones contain data and the status of interrupted task, in particular the address of the next microinstruction to be executed by the interrupted task. Select latches 11 also allocate priority to different tasks. The work store is addressed by means of a work store address register 12 (WSAR). An address may be entered into WSAR 12 directly from CDR or console switches 13. The auxiliary store 9 is addressed using auxiliary store address register 14 (ASAR). An address may be entered into ASAR 14 directly from CDR or by means of a digit from work store. Both WSAR and ASAR can be incremented.

Data is transmitted to the various I/O devices and to central processor unit (CPU) external latches 15 by Bus Out 16. The CPU external latches store information required for program execution for example masks, keys, interruption and instruction length codes, and status bits. Data is transmitted to the CPU from the I/O devices and the CPU external latches 15 by means of bus in 17. The CPU external latches can be set or examined by the microprogram.

Initially the microprogram is loaded into the control store by a microprogram load file 18. Each location in the storage unit 1 has error correction code bits (ECC) for correcting errors in words accessed from storage. Each

storage word has 64 data and 8 ECC bits. However, only 16 bits and 2 parity bits are transmitted for each storage operation.

The apparatus is also provided with control logic 19 which consists of a number of system control latches which duplicate certain bits of the PSW, etc., and are set when data is read into control store. The system control latches continuously provide signals on lines 20 to various parts of the data processing apparatus for testing for microprogram interrupts, for log out and error conditions; and also for testing for program interrupts.

The microinstructions are classed into five classes: storage, operate, external, branch and special.

Storage microinstructions are used for transferring data between work store and either control store or main store. A storage microinstruction may be indirect or direct. Indirect microinstructions use an address of the required storage location from work store which may be incremented or decremented by the microinstruction. Direct microinstructions use an address provided by the microinstruction and can only address a fixed area of the control store.

Operate microinstructions relate to the operation of the ALU. Data is processed by the ALU and the result is placed in work store or data is transferred from work store to auxiliary store through the ALU. An operate microinstruction may be a Work-Work, Work-Immediate, or Work-Auxiliary indirect or direct, the work-work microinstruction reads two operands from work store and places the result in work store. Work-immediate microinstruction used immediate data in the microinstruction with an operand from work store. Work-auxiliary microinstruction obtains one operand from work store and one from the auxiliary store, if indirect the address of the operand in the auxiliary store is a digit obtained from work store, if direct the address is specified by the microinstruction and the operand is located in the CWR of the auxiliary store.

External microinstructions include work-external and external-immediate. Work-external microinstruction transfers data between work store and an I/O device. The external-immediate microinstruction sets data specified in the microinstruction into an adapter of an I/O device.

Branch microinstructions are conditional, unconditional, multiway, or branch and link. Conditional branches test for a zero or non-zero bit in work store or in the CWR area of auxiliary store, or in the fixed area of control store or for a non-zero digit in work store. An unconditional branch replaces bits of the current microinstruction address in SAR. Multiway branch microinstructions may be 16-way depending on a digit in work store or 4-way using two low order bits of a digit in work store. The branch and link microinstructions replace bits of the current microinstruction address and store the return address, current address + 2, in a specified location in work store.

The special microinstructions are used for instruction fetch (I-fetch), for returning to a routine after a branch and link microinstruction, and for interrupts in which a new zone of work stork is selected and the return address is stored in the old zone of work store.

The fixed area of the control store which is addressable by work-storage direct microinstructions is divided into a common area of 64-halfwords (16 bits) which can be used by all I/O adapters and the CPU, and six private areas one for each of the six I/O adapters, e.g. low speed I/O printer, communications, disc file and two selector channels.

The common area stores portions of the PSW, various control registers including registers for program event recording (PER), time-out counts for I/O devices, log out data, a buffer for PER, interrupt data, and dumps and tables used in operating the data processing apparatus.

The CPU external registers 15 provide various controls for the CPU and console. There are two arrays of latches; a work-external array which consists of 16 half words and which has read/write controls to work store, and an external-immediate array which consists of 16-bytes (8 bits) and which can be set or reset using immediate data forming a part of an external immediate microinstruction.

Generally a particular control is present in only one of the arrays. However, some such as a condition code are present in both arrays, e.g. the two bit condition code in the PSW needs to be read into work store for branch instructions but must also be capable of being set using immediate data.

In operation the PSW contains all the information for controlling program execution. The PSW includes the instruction address, condition code, interruption code and other fields. A fuller description of the PSW is given in the IBM System 360 Principles of Operation (IBM is a Registered Trade Mark). In general the PSW is used to control instruction sequencing and to indicate the status of the system in relation to the program being executed. The active or controlling PSW is called the "current PSW". By storing the current PSW during a program interruption, the status of the processing apparatus can be preserved for subsequent use. By loading a new PSW or part of a PSW, the state of the processor can be initialized or changed.

An instruction consisting of 1, 2, or 3 halfwords is fetched under the control of an I-fetch microinstruction in the CDR. The instruction fetch operation depends on the format of the instruction. Instructions have RR, RX, RS, SI and SS formats and are des-

cribed in detail in the above referenced Principles of Operation. An RR format relates to a register to register operation, i.e. two operands from the auxiliary store. An RX format relates to a register to indexed storage operation. An RS format relates to a register to storage operation. An SI denotes a storage and immediate operand operation. An SS format relates to a storage to storage operation.

The instruction count or address is stored in the CPU zone of work store. The CPU zone has eight addressable words of storage. The locations are allocated as follows:—

Locations 0 and 1—Work Area
Locations 2 and 3—Effective Address and Work Area
Locations 4 and 5—Instruction Count
Location 6—First Halfword of Instruction
Location 7—Microprogram Start Address.

As stated above an instruction is fetched from main storage under the control of the I-fetch microinstruction using an address provided by the instruction count in work store locations 4 and 5. The count in locations 4 and 5 is then incremented by 2. The first half-word of the instruction is set into location 6 and the instruction code is examined to determine the format of the instruction.

If the instruction has an RR format, the second operand is obtained from a GPR in the auxiliary store by gating the R_2 address field of the instruction to ASAR directly from location 6 in work store. Operand 2 is loaded into work store locations 0 and 1, or 2 and 3. The execution of their instruction can now begin.

If the instruction has a format other than RR, a further access is made to main storage using the address in locations 4 and 5 of work store which are given a further increment of 2. The second half word is set in location 3 of work store. The B_2 field in the second half-word of the instruction is used as an address for a GPR. The contents of this GPR is added to the D_2 field of the second half-word of the instruction in location 3 and the sum $B_2 + D_2$ stored in locations 2 and 3 of work store. If the instruction has an RX format the contents of the GPR indicated by the X_2 field is added to the contents, $B_2 + D_2$, of locations 2 and 3 and the sum stored in locations 2 and 3. Operand 2 is then fetched from main store at an address indicated by the contents of locations 2 and 3. Execution can now commence for RX, RS and SI instructions.

If the instruction has an SS format a further access to main store is necessary to fetch the third half-word of the instruction. Two main store addresses have to be calculated and two operands fetched from storage before the instruction can be executed.

A typical sequence of microinstructions for fetching the instruction and operands, executing the instruction, and storing the result is as follows:

RX Add

I-fetch
↓
Branch (Index)
↓
Branch (Instruction)
↓
Fetch Operand 2
↓
Branch (Operation code)
↓
Add operand 1 and 2
↓
Branch (overflow)
↓
Store result in GPR
↓
Branch (sign)
↓
Branch (zero)

Having explained the operation of the data processing apparatus for one instruction, it will be clear that a program of instructions can be executed in the same way. As described above to assist the programmer in debugging programs, a program event recording feature (PER) is incorporated which permits the program to be alerted for certain predetermined events. Four typical events outlined above are:—

1. the fetching of an instruction from designated main storage locations;
2. the successful execution of a branch instruction;
3. the alteration or use of designated general registers (GPR);
and
4. the alteration or use of designated main storage (MS) locations.

The program has control over conditions that are considered program events (PE) for recording purposes and can specify selectively one or more events to be monitored. The information concerning a program event is provided to the program by means of a program interruption with the cause of the interruption identified in the interruption code field of the PSW. The PE information is stored in three control registers A, B and C located in the common area of control store.

The format of the registers A, B and C is as follows:—

Control Register A

PER CODE		PER — GPR ALTERATION CODE BITS
0 7	8 15	16 31

Control Register B

	PER STARTING ADDRESS
0 7	8 31

Control Register C

	PER ENDING ADDRESS
0 7	8 31

PER code bits 0 to 7 of control register A indicate the events which are to be detected and recorded. The bits are assigned as follows:—

Bit 0—Successful Branch Code bit
Bit 1—Instruction Fetch Code bit
Bit 2—Main Storage Code bit
Bit 3—GPR Code bit
Bits 4 to 7—Other Events

It is clear that events other than the four defined above can be recorded and that bits 4 to 7 or different combinations of bits can be used to specify these other events. A one on any one of portions 0 to 3 indicate that the event is to be recorded. A zero causes the event to go unrecorded.

PER—GPR Alteration code bits 16 to 31 of register A specfiy which of the GPR's are to be monitored for alteration or use of their contents. Each of the 16 bits refers to a different one of the 16 GPR's located in auxiliary store 9 in the order of ascending addresses. When a bit is a 'one', the register is included, when 'zero' the register is excluded from PER.

PER starting address bits 8 to 31 of control register B an address that designates the beginning of a monitored main storage area. PER ending address bits 8 to 31 of control register C designates the end of the monitored area of main storage.

In addition to the PER code bits, a program event mode bit in the PSW may be used to control PER, e.g., if the PE mode bit in the PSW is 'one' and a PE code is 'one' then PER is active; but if the PE mode bit is zero PER is inhibited.

Program interrupts for PER are handled in the same way as any other interrupt. Control logic 19 is provided with a number of extra latches for PER. A PE mode latch is provided for a PE mode bit and PE code latches are provided for the PE Code bits. A PE detected latch is provided for indicating that a program interrupt is to be taken to record the detected event and a PE Log latch is provided for handling certain more complicated events as will be described later.

The PE code bits need not be held in system control latches in control logic 19 but could be held in the CPU external latches 15 in which case the PE code bits are tested by the microprogram.

A particular embodiment of the invention will now be described with reference to the Flow Chart which shows the detection and recording of each of the four events described above. However, it will be appreciated that other events may be detected and recorded and that other types of data processing apparatus can be used.

Refer to sheet A of the flow chart which starts with the I-fetch microinstruction (50) in the CDR. (N.B. the numbers in parenthesis refer to points in the flow chart). As part of the I-fetch microinstruction a number of tests are performed by control logic 19. For PER, the PE detected latch (51) is tested and if set a program interrupt is taken to record an event which had been detected during the last instruction execution. The setting of the PE detected latch and the program interrupt will be described later.

If the PE detected latch (51) is not set, the PE mode latch (52) is tested. If the PE mode latch is not set program event recording is inhibited, the I-fetch operation is completed and the microprogram continues to the

branch on operations code microinstruction (53), i.e., the operations code of the fetched instruction. Tests (51) and (52) occur simultaneously and may be combined by suitable logic.

The branch (54) sheet B is performed by forcing an address in SAR of the next microinstruction. The address depends on bits 0 and 1 of the operation code, i.e., RR; RX; RS or SI; or SS formats. Also at this point a microprogram interrupt can be taken to select a different zone of work store.

A program event is detected if an appropriate microinstruction is executed and an associated PE code bit or latch is set.

At I-fetch if the event mode latch is set a test (55) is made to determine whether the I-fetch address compare code bit is set. Test (55) may be performed by the microprogram testing the code bits in external latches 15. However by providing latches in control logic 19 for the PE code bits, the test (55) may be performed by the hardware system control logic by comparing the PE code bits and bits of the I-fetch microinstruction. Tests (52) and (55) may be combined by using suitable logic.

If the PE I-fetch address compare code bit is not set (test 55) the main microprogram continues. If the PE I-fetch code bit is set, the microprogram tests (56) whether the address of the instruction is between the PER starting and ending addresses. Test (56) can be performed by comparing the addresses in control registers B and C with the instruction count. If the instruction address is not between the PER starting and ending addresses, the microinstruction "complete I-fetch" (57) microinstruction returns the operation to the main microprogram which proceeds to the branch on operations code microinstruction (53). The "complete I-fetch" microinstruction allows I-fetch to be executed without testing for a program event. If the instruction address is in the range between starting and ending addresses, a program event is detected and the microprogram sets (58) the PE code and the address of the instruction in a buffer register in the common area of control store. The microprogram then sets the PE detected latch (59). At this point the "complete I-fetch" microinstruction (57) returns operation to the main microprogram.

Assuming that no program events are detected after the branch on operations code microinstruction 53, the microprogram returns to point A and a further I-fetch (50) is made. During this I-fetch test (51) is made as before to determine if an event has to be recorded. Assuming that the PE detected latch had been set during the previous I-fetch, a program interrupt is taken to record the event.

During the interrupt the program event code and the address in the buffer are transferred (60) and (61) to specified locations in main store. When an instruction produces more than one event all the events are concurrently indicated by the PE code i.e., more than one PE code bit may be present but only one instruction address is needed. The program interruption is always taken after the execution of the instruction responsible for the event so that the occurrence of the event does not affect the execution of the instruction, which may be completed, terminated or suppressed.

During the program interrupt the current PSW is updated (62), i.e., the address of the next instruction is calculated and inserted in the instruction count field of the PSW and a PE interrupt code is inserted in the interrupt field of the PSW to indicate the cause of the interrupt.

The PE detected latch and the buffer are then reset (63). The current PSW is stored in main store (64) and a new PSW is loaded (65) for analysis of the interrupt by a program subroutine (66). After analysis of the interrupt the old PSW is usually reloaded by the programmer (67) and operation returns to A where I-fetch is continued. When more than one PE code bit is present indicating that more than one event has taken place, only one interrupt is required as the interrupt analysis is designed to handle multiple events.

The other program events can be detected after the branch on operations code microinstruction (53) as shown on sheets B and C of the flow chart. These events may be detected when certain microinstructions are used, i.e. a GPR alteration or use event can be detected when a GPR write instruction is used, a main storage alteration or use event can be detected when a M.S. write microinstruction is used and a successful branch event can be detected when a successful branch microinstruction occurs. The successful branch microinstruction always follows a successful branch in the microprogram used in this particular embodiment.

A GPR or MS write microinstruction, or a successful branch microinstruction can occur in a number of the microprogram routines which follow the branch on operation code, but only one example of each is shown on the flow chart.

The successful branch event is shown on sheet B. A successful branch instruction is always terminated by successful branch microinstruction (68). As described for the I-fetch event the event mode latch is tested (69). If the event mode latch is not set, the microprogram returns to A and continues with the next I-fetch microinstruction. If the event mode latch is set, a test (70) is made by microprogram or control logic to determine if the successful branch PE code bit is set. As stated above test 70 can be performed by comparing the PE code bits with bits of the microinstruction. If the test (70) is negative the microprogram returns to A. If the test (70) is

positive, a program event is detected, and the PE code and address of the branch instruction is stored in the buffer (71). As the branch instruction has already been executed, the program interrupt can be taken immediately and the operation proceeds to entry B on sheet A. For this event there is no need to set the PE detected latch to provide the interrupt. The interrupt sequence (60) to (67) is followed as described above for the I-fetch address compare program event.

The GPR and MS alternation or use program events are shown on sheets B and C of the flow chart. The events are detected when a write microinstruction is used. These events can occur during the execution of instructions with a number of formats and are also more complicated to detect.

When a GPR write (72) or a MS write (73) microinstruction is loaded into the CDR, the PE mode latch (74) or (75) is tested. Test (74) or (75) may be combined with the testing of the GPR or MS PE code bit (76) or (77) respectively. As described above tests (76) and (77) may be made by microprogram or by the control logic 19 in which case the tests (74) to (77) may be carried out by suitably designed logic.

If the event mode latch is not set or if the appropriate PE code bit is not set, the main microprogram is continued (78) or (79) and the microprogram returns to A. If the test (74) or (75) indicates that the event mode latch is set and either of the tests (76) or (77) indicate that the appropriate PE code bit is set a possible program event is detected and a PE log latch is set (80). The PE log latch is located in control logic 19 and is set to enable information concerning the program event to be logged-out to the console zone of work store (82). The procedure allows the data logged out to be analysed without disturbing the CPU zone of work store which was active when the possible event was detected. The data is logged-out to the console zone under control of control logic 19. Log-out logic is primarily provided for handling error condition program checks, for instruction retry and for system reset (81). The data logged-out to the console zone includes the contents of SAR, BAR, ASAR, WSAR and the CPU zone of work store. Note that the analysis of an error or check condition in console zone takes priority over the analysis of a program event.

Operation of the data processing apparatus now continues in console zone of work store by forcing a microprogram start address to a microprogram routine for analysing the program event (83). The microprogram now tests if the PE log latch has been set (84). If the PE log latch has not been set the microprogram enters an error handling or a system reset subroutine (85). If the PE log latch is set, the microprogram commences the analysis of the program event.

The microprogram now tests (86) if the detected event is a GPR or a Main store event (sheet C). This test can be carried out by testing a particular bit position of the microinstruction which was in the CDR when the possible event was detected. This microinstruction was read from control store using the contents of BAR now in console zone.

If the possible event is a GPR event, the microprogram tests (87) if the contents of ASAR matches the GPR alteration mask bits in register A located in control store. If the possible event is a MS event, the microprogram tests (88) if the logged-out SAR address, is between the PER starting and ending addresses. If the result of test (87) or (88) is positive, a program event is detected, and the PE code and address in Main store or ASAR are stored in the buffer in control store (89). The PE detected latch is now set (90). The setting of the PE detected latch enables a program interrupt to be taken after the current instruction has been executed.

The microprogram continues after the PE detected latch has been set or if tests (87) or (88) prove negative. The PE log latch (91) is reset and the CPU zone of work store selected (92). The execution of the current instruction now recommences (93) using a microprogram start address in the CPU zone of work store. Eventually the operation returns to A where the next I-fetch is attempted or an interrupt taken if the PE detected latch is set.

For the GPR and MS program events the log-out facility which is primarily provided for error handling program checks, instruction retry or system reset can be used. Accordingly, it will be appreciated that the present invention may be embodied in a data processing apparatus without requiring the addition of more than a few latches carrying out certain of the program event tests, the remaining tests being carried out by the microprogram. The hardware latches ensure that the program event microprogram is entered only if the appropriate PSW and control register mode bits are set on. Thus there is no degradation in CPU performance when the programmer wishes to ignore program event recording. By using the microprogram to carry out the various tests the use of additional hardware latches can be avoided.

WHAT WE CLAIM IS: —

1. Data processing apparatus for executing a program of instructions, in which predetermined situations occurring during the execution of an instruction are designated program events, the apparatus including means for indicating that a particular program event or events is to be detected, means for detecting that an indicated event has occurred, means responsive to the detection of an indicated event for recording that the indicated event has occurred, means for completing the execution of the instruction during which the indicated

program event was detected, and microprogram control means for fetching instructions from storage and for executing instructions in arithmetical and logical unit, in which each program event has an associated microinstruction, and in which the detecting means is responsive to the indication that a particular event is to be detected and to the occurrence of the microinstruction associated with the event.

2. Apparatus as claimed in claim 1, including means for generating a program interruption on completion of the execution of the instruction in which an indicated event was detected whereby the recording of the event can be delayed until that instruction has been executed.

3. Apparatus as claimed in claim 2, in which the interruption generating means includes a program event detected latch settable by the detecting means in response to the detection of an indicated event.

4. Apparatus as claimed in claim 2 or 3 including a buffer for storing a code representing the event or events detected and information indicating the storage or register location at which the event occurred or the address of the instruction in which the event occurred; in which the recording means is effective during the program interrupt for transferring the program event code and the information stored in the buffer to specified main storage locations.

5. Apparatus as claimed in claim 3, in which the program event detected latch is tested at the time each instruction is fetched from storage and in which if the program event detected latch is set during the execution of the previous instruction the program interruption is taken to record the event.

6. Apparatus as claimed in any one of the preceding claims including means for inhibiting operation of the program event indicating means, whereby the apparatus executes instructions without recording program events.

7. Apparatus as claimed in claim 6, in which the inhibiting means includes a program event mode indicator.

8. Apparatus as claimed in claim 7, in which the program event mode indicator is provided by a bit of a program status word.

9. Apparatus as claimed in claim 2 and any one of claims 3 to 8 including means for providing in a program status word a program interrupt code indicating that the interrupt is required for recording a program event.

10. Apparatus as claimed in any one of the preceding claims including means for providing a log-out of data from operating parts of the apparatus for handling errors, checks and instruction retry, in which the detecting means uses said means for providing a log-out of data when a program event requiring analysis is indicated; whereby data in the operating parts of the apparatus relating to the execution of the current instruction is preserved, so that execution of the current instruction can be completed after the indicated event has been analysed.

11. Apparatus as claimed in claim 1 including a plurality of latches, in which each latch indicates that a different event is to be detected and in which the outputs of the latches are logically combined with particular bits of the microinstructions for detecting the occurrence of an event.

12. Apparatus as claimed in claims 7 and 11 including a program event mode latch settable in accordance with the program event mode indicator, in which the output of the program event mode latch is logically combined with the outputs of the plurality of latches whereby the program event indication given by the plurality of latches may be inhibited.

13. Apparatus as claimed in claim 9 including a local store having one portion assigned for operations involving instruction fetch and execution, and another portion primarily assigned for operations involving the log-out of data from operating parts of the apparatus; in which in the detection of a program event requiring analysis, data relating to the program event is logged-out to said other portion of the store while data relating to the execution of the current instruction is preserved in said one portion of the local store.

14. Apparatus as claimed in claim 13, in which said one portion of the local store stores a microprogram address from which execution of the current instruction can recommence after the detected program event has been analysed in said other portion of the local store.

15. Apparatus as claimed in claim 13 or claim 14 including control logic for controlling log-out of data to said other portion and for forcing a microprogram start address whereby the microprogram analyses the detected program event.

16. Apparatus as claimed in claim 3 and any one of claims 13, 14 or 15, in which the program event detected latch is set when the analysis of the event satisfied a predetermined condition.

17. Apparatus as claimed in any one of claims 13 to 16, in which a first and a second program event relate to the alteration or use of a particular local register, and to the alteration or use of a main storage location within a particular range of locations, respectively; in which a local register microinstruction and a main storage microinstruction are associated with the first and second program events respectively; in which the data logged-out to said other portion of the local store includes the contents of the main storage address register, the local register address and the contents of said one portion of the local store; and in which the analysis includes a comparison of

the address of the particular local register or particular range of locations with the address of the actual register or the actual location respectively.

18. Apparatus as claimed in claim 11 or 12, in which a third program event relates to the fetching of an instruction from a designated area of storage and is associated with instruction fetch microinstruction and in which the microprogram compares the actual address of the instruction with the range of address of the designated area of storage.

19. Apparatus as claimed in claims 3 and 18, in which the program event detected latch is set when the actual address is within the range of address of the designated area of storage.

20. Apparatus as claimed in claim 11 or 12, in which a fourth program event relates to the successful execution of a branch instruction and is associated with a successful branch microinstruction which follows the successful branch.

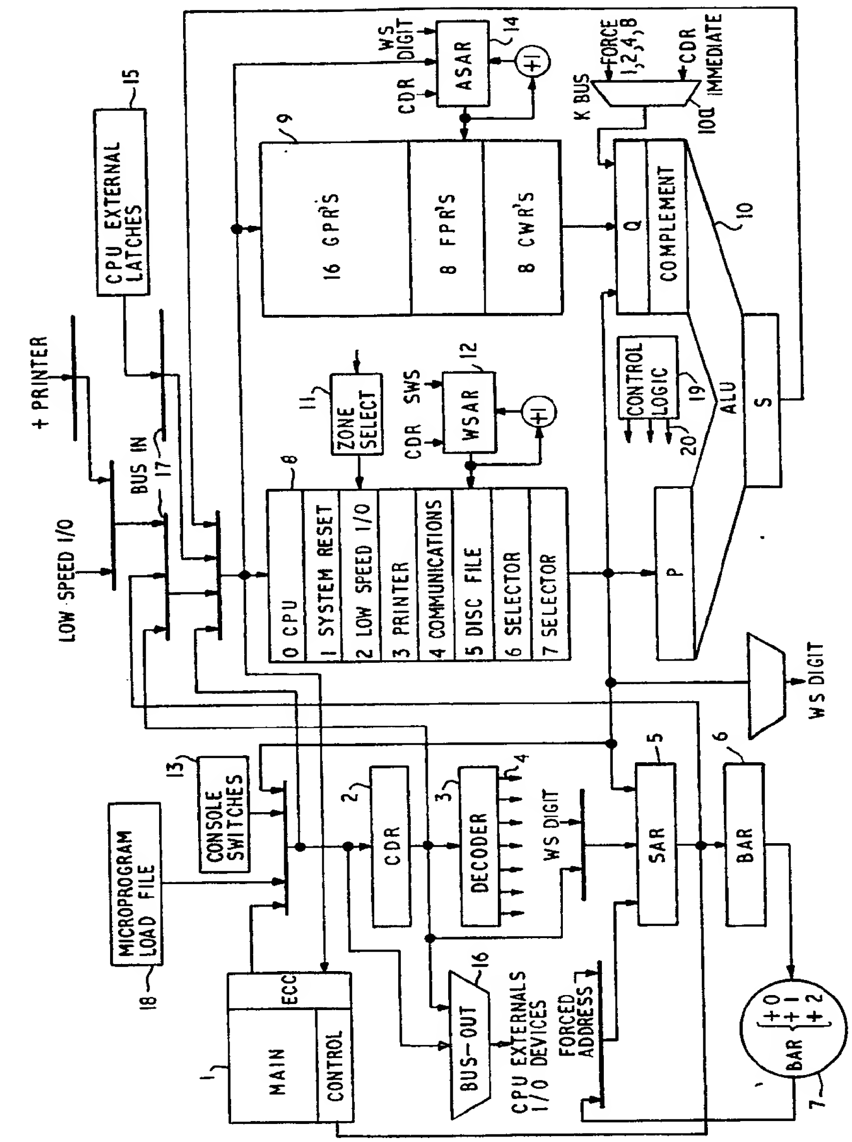
21. Apparatus as claimed in claims 3 and 20, in which the program interrupt is taken immediately the fourth program event is detected, without setting the program event detected latch.

22. Data processing apparatus substantially as described with reference to the drawing.

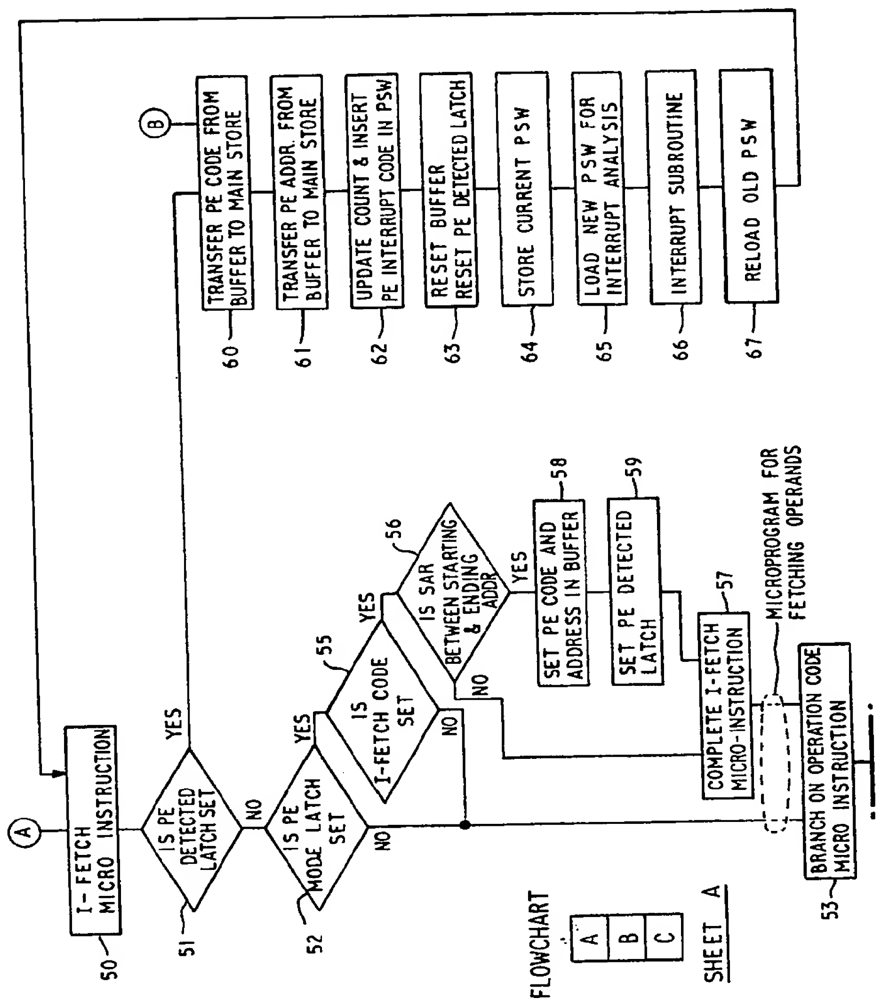
23. Data processing apparatus substantially as described with reference to the flow chart.

ALLAN E. MITCHELL,
Chartered Patent Agent,
Agent for the Applicants.

Printed for Her Majesty's Stationery Office by the Courier Press, Leamington Spa, 1972.
Published by the Patent Office, 25 Southampton Buildings, London, WC2A 1AY, from which copies may be obtained.



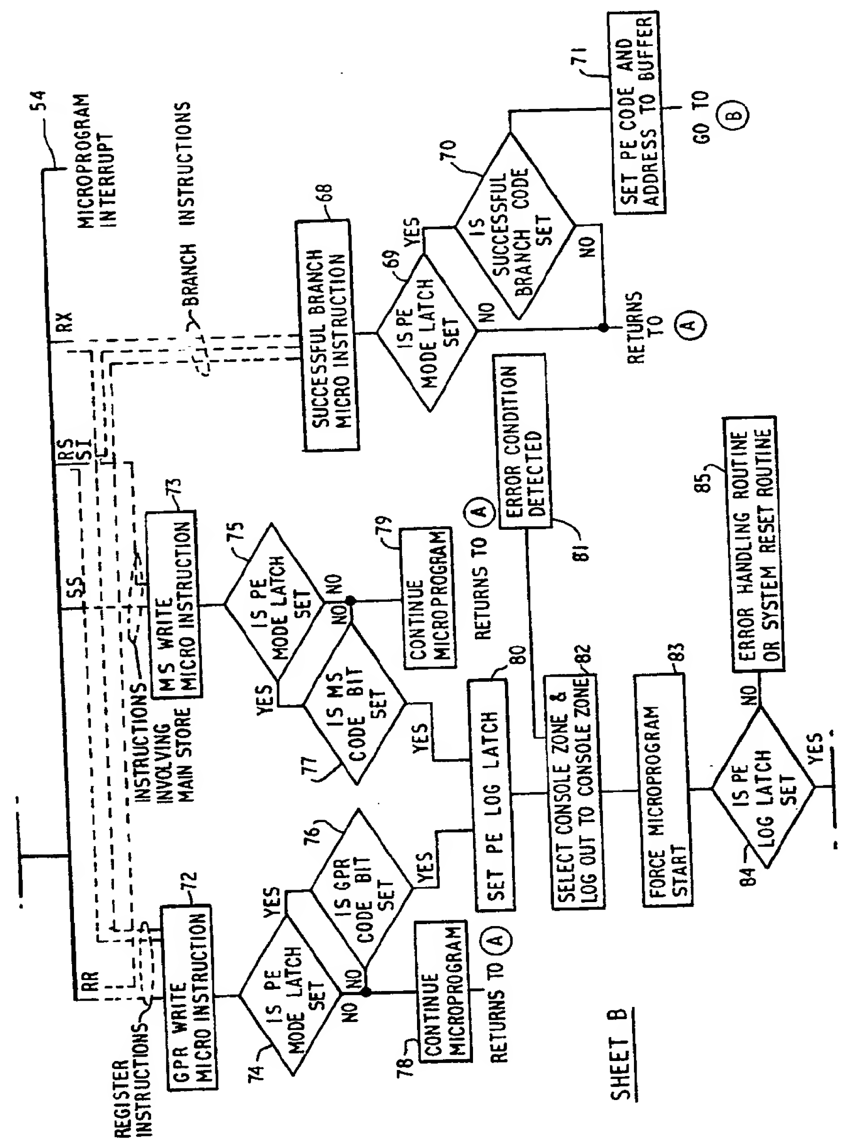
+ PRINTER
LOW SPEED I/O
BUS IN
17
CPU EXTERNAL LATCHES
15
16 GPR'S
8 FPR'S
8 CWR'S
9
WS DIGIT
CDR
ASAR
14
+1
K BUS
FORCE 1,2,4,8
CDR IMMEDIATE
10a
Q
COMPLEMENT
10
ZONE SELECT
11
CDR
SWS
WSAR
12
CONTROL LOGIC
19
20
ALU
S
P
8
0 CPU
1 SYSTEM RESET
2 LOW SPEED I/O
3 PRINTER
4 COMMUNICATIONS
5 DISC FILE
6 SELECTOR
7 SELECTOR
WS DIGIT
13
CONSOLE SWITCHES
MICROPROGRAM LOAD FILE
18
2
CDR
3
DECODER
4
WS DIGIT
5
SAR
6
BAR
MAIN
ECC
CONTROL
1
BUS-OUT
16
CPU EXTERNALS
I/O DEVICES
FORCED ADDRESS
BAR
+0
+1
+2
7

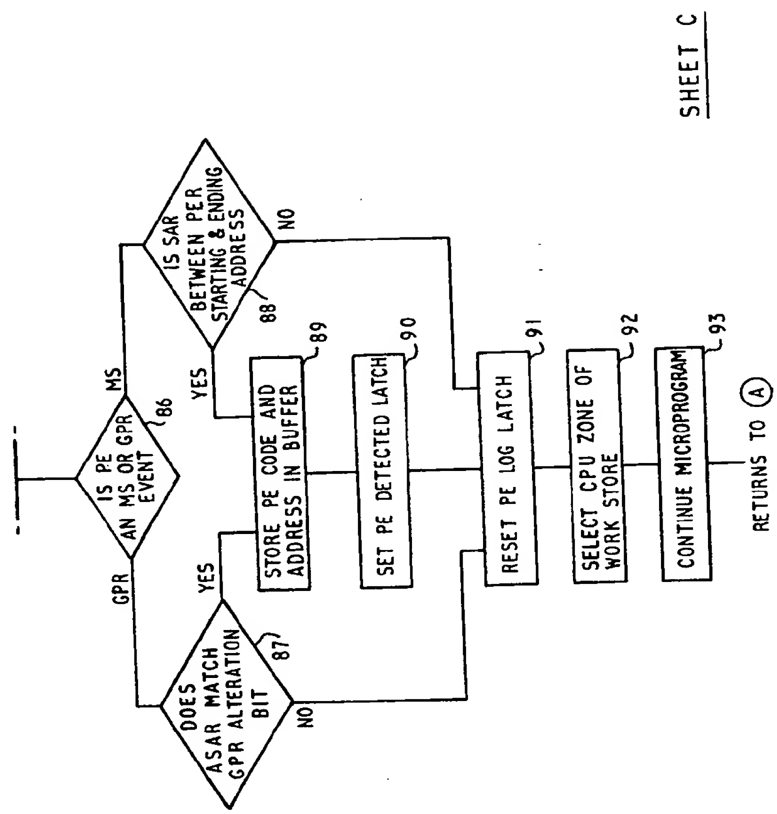

A
I-FETCH MICRO INSTRUCTION
50
IS PE DETECTED LATCH SET
51
YES
NO
IS PE MODE LATCH SET
52
YES
NO
IS I-FETCH CODE SET
55
YES
NO
IS SAR BETWEEN STARTING & ENDING ADDR
56
YES
NO
SET PE CODE AND ADDRESS IN BUFFER
58
SET PE DETECTED LATCH
59
COMPLETE I-FETCH MICRO-INSTRUCTION
57
MICROPROGRAM FOR FETCHING OPERANDS
BRANCH ON OPERATION CODE MICRO INSTRUCTION
53
B
TRANSFER PE CODE FROM BUFFER TO MAIN STORE
60
TRANSFER PE ADDR. FROM BUFFER TO MAIN STORE
61
UPDATE COUNT & INSERT PE INTERRUPT CODE IN PSW
62
RESET BUFFER RESET PE DETECTED LATCH
63
STORE CURRENT PSW
64
LOAD NEW PSW FOR INTERRUPT ANALYSIS
65
INTERRUPT SUBROUTINE
66
RELOAD OLD PSW
67
FLOWCHART
A
B
C
SHEET A

1264066 COMPLETE SPECIFICATION

4 SHEETS *This drawing is a reproduction of the Original on a reduced scale*

Sheet 3




SHEET C
IS PE AN MS OR GPR EVENT
86
MS
GPR
IS SAR BETWEEN PER STARTING & ENDING ADDRESS
88
YES
NO
DOES ASAR MATCH GPR ALTERATION BIT
87
YES
NO
STORE PE CODE AND ADDRESS IN BUFFER
89
SET PE DETECTED LATCH
90
RESET PE LOG LATCH
91
SELECT CPU ZONE OF WORK STORE
92
CONTINUE MICROPROGRAM
93
RETURNS TO A